Impact of Installation Counts on Perceived Quality: A Case Study on Debian

Israel Herraiz
Technical University of Madrid, Spain
israel.herraiz@upm.es

Emad Shihab, Thanh H.D. Nguyen, Ahmed E. Hassan
Software Analysis and Intelligence Lab
Queen's University, Canada
{emads, thanhnguyen, ahmed}@cs.queensu.ca

***Abstract*—Software defects are generally used to indicate software quality. However, due to the nature of software, we are often only able to know about the defects found and reported; either following the testing process or after being deployed. In software research studies, it is assumed that a higher amount of defect reports represents a higher amount of defects in the software system. In this paper, we argue that widely deployed programs have more *reported* defects, regardless of their *actual* number of defects. To address this question, we perform a case study on the Debian GNU/Linux distribution, a well-known free / open source software collection. We compare the defects reported for all the software packages in Debian with their popularity. We find that the number of reported defects for a Debian package is limited by its popularity. This finding has implications on defect prediction studies, showing that they need to consider the impact of popularity on perceived quality, otherwise they might be risking bias.**

***Index Terms*—defects; quality; popularity; Debian**

I. Introduction

Software defect prediction is an emerging area of research, where studies attempt to establish a relationship between a set of factors (e.g., size) and software defects. The main motivation behind software defect prediction is assisting practitioners identify the defect-prone locations of a software system. The nature of the factors that are believed to predict defects is diverse. For example, prior work investigated the impact of static metrics [6], [9], [20], process metrics [10] and people-related metrics (e.g., developer experience) [8] on software quality. Due to the availability of data in free / open source software, many of these studies are based on defects data extracted from free / open source projects [6], [10], [20].

However, what most of the previous work actually predicts are *reported* defects. In fact, in most large software systems it is extremely difficult to know *all* of the defects in the system without extensive (and practically infeasible amounts of) testing. Therefore, research studies are based on found defects, i.e., on perceived quality.

We hypothesize that *widely deployed systems will have more reported defects, regardless of their actual quality*. To investigate our hypothesis, we use the Debian GNU/Linux distribution, a collection of software packages. For every package in Debian, it is possible to know the defect reports associated to that package, and the number of people who installed it (and voluntarily reported it). We evaluate how defect reports and installation counts are related, and what how other factors such as package age, importance and profile of the reporter (user or developer) affect the relationship between defect reports and installation counts.

Our findings indicate the following:

- The number of defects reported for a Debian package is limited by its installation counts.
- The age of a package, type of package and its installation priority do not influence the relationship between installation counts and number of reported defects.
- The reporter of a defect report (whether Debian developer or not) does not influence the relationship between the installation counts and defect reports.

The rest of the paper is organized as follows. Section II highlights the background and related work. Section III explains the data source used in this paper, and the software quality process in Debian. Section IV details our methodology. Section V presents our results. Section VI lists the threats to validity and Section VII concludes the paper.

II. Background and Related Work

Debian has been a case study for empirical software engineering research in previous occasions. Its evolution has been studied as a particular example [17], because it is a large software compilation driven by processes different to those found in software development projects. The data sources that are available in Debian have been structured into a database for research purposes: the *Ultimate Debian Database* [14]. This database has already been used for software defect research by Davies *et al.* [4], again focusing on the differences between Debian as a large software compilation, compared to a software development project. The main difference in what regards to defects is that Debian defects are not associated to commits in a version control system, but to packages that are uploaded to a repository when a new defect has been fixed. Davies *et al.* are the first ones to suggest a relationship between popularity and defects, but they could not find evidence in their data to support that claim.

Because of the availability of data, other free / open source software projects have been used as case studies for software defects research. Mockus *et al.* [11] found that the (reported) defects density of Mozilla and Apache was lower than other non-open source software projects, concluding that it was an evidence of the high quality of open source software development. Paulson *et al.* [15] also studied a set of open and

closed source projects, concluding that open source software contained less (reported) defects because they were found and fixed rapidly. Most studies (included the two aforementioned) assume that a higher number of reports necessarily mean a higher number of actual defects (or the other way around, that a lower number of reported defects means a lower number of actual defects).

Other studies based on free / open source software investigate the impact of different factors on software defects. Gyimothy *et al.* [6] use several product metrics to predict fault-proneness using different statistical methods. Using metrics as defect predictors is of course not a new idea; as an example of this prolific line of research, we can cite this paper by Fenton and Ohlsson [5], and another by Zimmermann *et al.* [20], where the authors created a dataset for research purposes containing defects counts and code complexity metrics for all the modules of Eclipse. In another example, Menzies *et al.* [9] presented defect prediction models based on static attributes such as size and complexity.

Some approaches are based on the dependencies between modules in the system [19], and the relationships between developers [8]. This line of works is based on the idea that defects tend to be related to violated dependencies in the system; Cataldo *et al.* [3] found that dependencies networks have more influence in fault proneness than developers networks, and that developers networks than syntactic networks.

However, none of the studies mentioned here consider the impact of usage patterns and deployment issues on quality as perceived by users. To our knowledge, the only paper to address the impact of usage patterns is Mockus *et al.* [12], where they investigated the impact of usage patterns and deployment issues on customer perceived quality for a set of industrial (non-open source) projects. They found that factors such as amount of usage, hardware configurations, software platform and deployment time could affect the probability of observing a defect. However, they do not consider the number of installations (or the number of users) as one of those factors. In any case, they also remark the influence of usage patterns and deployment issues on software defects, and the risk of bias on defects prediction if these factors are ignored.

Bias in software quality data has also been discussed in recent studies. Recent studies [1], [2], [13] have shown that linkage and tagging bias might affect the results of software quality studies.

III. Software Quality in Debian

In this section, we provide an overview of the software quality in Debian. First, we provide an overview of the Debian software distribution. Then, we describe the *Debian Popularity Contest* (Popcon) and its defect tracking system. Finally, we outline the differences between Source and Binary packages in Debian.

A. The Debian GNU/Linux distribution

Distributions are collections of software, where all the different programs are integrated, solving the problem of installing the necessary dependencies to install a particular software package. Distributions also make it easier to install the system from scratch, without having to get every part of the system (kernel, shell, utilities, desktop, etc) from their original (and dispersed) locations. These distributions retrieve the code from third party sites, adapt and integrate it with the rest of the distribution, and make them available to users in the form of *packages*.

In the case of Debian, every package is maintained by volunteer developers and maintainers who are in charge of retrieving the source code from the third party sites, and maintaining the packages, which usually contain defects reported both by users and developers.

Packages are individually installed by users of the distribution. Users can voluntarily opt-in to the *Debian Popularity Contest* (Popcon), which tracks the installation of all the packages, and make some metrics available about those installations (more details are given in the next subsection).

The advantage of Debian as a case study over other options is the availability of users' information through the Popcon. Other case studies leave traces about the development and maintenance process, but the users' side is virtually invisible because there are no repositories that record software installations. To the best of our knowledge, Debian is one of the few cases tracking this information (Ubuntu is another example) and makes it publicly available.

Regarding defects, Debian relies on the feedback provided by users and developers to improve the quality of the system. This feedback is done in the form of defect reports, with reports referring to one (or more) software package, with some minor exceptions (some reports are for internal maintenance activities, but those are discarded in this study). Besides that, the Debian defect tracking system is similar to other cases.

B. The Debian Popularity Contest

Debian has a user installations tracking system, that counts software downloads, installations and removal, called the *Debian Popularity Contest* (Popcon). Users can opt-in to send a report to the Popcon server every time they install or remove a package. Popcon statistics are used by Debian to prioritize decisions about software packages. Packages that are more important to its users get higher priority. For instance, packages in the Debian DVDs are sorted in the different disks by popularity. FTP mirrors also decide which packages to store depending on their popularity.

In Popcon, users can report in an automatic and non-intrusive fashion what packages they install (or remove). The reports are sent periodically to a server through email or HTTP. Then, the report aggregates the statistics for every package in the distribution. The available statistics are the following:

- **Installation**: The number of users who installed the software package. This is the sum of **Vote** and **Old**.
- **Vote**: The number of users who use the software package regularly (in the last 30 days).
- **Old**: The number of users who installed but did not use the software package regularly.

- **Recent**: The number of users who recently upgraded the software package (in the last 30 days).
- **No files**: The number of people whose entry did not contain enough information, e.g., because of an error in the data transmission.

C. The Debian Bug Tracking System

In Debian, defects are reported using a specific tool, known as `reportbug`, that fills the necessary fields and tags of each defect report. Then `reportbug` send each report to the *Debian Bug Tracking System*, where it is archived. The term used in the project to denominate a defect is *bug*, which is very usual between software developers.

Reports can be assigned to packages, or to other activities (e.g., a request for a new package to be added to the distribution). In this study, we only consider defects attached to packages. Because of the way `reportbug` works, it is not possible to have a defect report referring to a package that is not accordingly marked with the package name in the defect tracking system. Also a bug report can be attached to several packages.

The name and email address of the user reporting the defect is also recorded. Users can select the severity of the defect; this decision is assisted by the tool, which asks questions about the impact of the defect. Defects are automatically notified to the maintainer of the referred package, who can take ownership of the defect report and inquire for more information from the user, if needed.

The packaging process consists of two main stages, which are potential sources of defects in the packages:

- Source code retrieval from *upstream*.
- Source code modification to adapt it to the rest of the distribution.

In the Debian argot, *upstream* means the open source project from where the source code is taken. This open source project is usually a third party that is not related to Debian. In this packaging process, maintainers can unintentionally introduce defects, because they change the source code, move files to different locations, and in general adapt the source code to the packaging and installation standards of the distribution. When a defect is reported, the maintainer checks whether the defect is specific to Debian or not. If she thinks it is not related to Debian, she forwards the defect to upstream.

Thus, the result of a defect report can be:

- **Accepted**: This is a defect that has been confirmed by the maintainer of the package. The defect is specific to Debian, otherwise it would be forwarded. If the defect is fixed, it will be marked accordingly when closed.
- **Forwarded**: The defect has been confirmed by the maintainer of the package, but it is not specific to Debian. The defect is then forwarded to *upstream*. The defect is not followed in the Debian defect tracking system. If upstream releases a new version of the source code, it will be imported when the Debian package is updated.
- **Rejected**: The maintainer cannot confirm the defect, either in their own package or in the original pristine upstream source code. The report is rejected, and the user can decide to submit it again providing more details to reproduce the defect.

D. Source and Binary Packages

There are two kinds of packages in Debian: binary and source. A source package will generate one or more binary packages. Thus, the number of binary packages is always greater than or equal to the number of source code packages.

Since users install binary packages, Popcon statistics are collected at the binary level. However, defects are associated to source packages, because it is source packages where the source code is found, and it's a modification in the source package that will fix the defect. Thus, we need to reconcile binary and source packages to cross correlate data from the defect tracking system with the popularity of packages.

With regards to Popcon statistics, the Ultimate Debian Database [14] deals with this issue using two approaches: **Maximum popularity**, the popularity of a source package is the maximum popularity of the generated binary packages; and **Average popularity**, the popularity of a source package is the average popularity calculated using all the generated binary packages.

Both values will be the same only if the source package generates one binary package (or if all the binary packages have the same popularity). We use the first approach, which assumes that if a user has installed at least one of the binary packages corresponding to a source package, she is also a user of the source package. This is a fair assumption, because if she has installed only one binary package of a source package, and she reports a defect in that binary package, the report will be filled against the corresponding source package.

Source packages contain some meta-information that is useful to classify them. Among the different meta-data, are the section, which allows us to classify the package as a library or as a standalone application; the priority, which marks the package as essential or optional. Packages that are essential will irremediably have a high value of popularity, because all users reporting to the Popcon will have installed them.

IV. Methodology

In this section, we define the metrics used in this study, and explain how the data was retrieved and prepared for the statistical analysis.

A. Definitions and Metrics

For our study, we analyze the relationships between popularity (i.e., number of installations) and defects, splitting the analysis using different criteria such as the age of the packages, and the type of application. Here, we define the different metrics and parameters used for the study.

1) Popularity and Defects: Let s be a source package. This package can be installed in the system through a set of different binary packages:

$$s = \{b_i\} \tag{1}$$

Popularity is defined for a binary package. We assume that the popularity of a source package is the maximum popularity of the corresponding binary packages. Let p be the popularity

$$p_s = \max(\{p_{b_i}\}) \tag{2}$$

Popularity can be defined through different metrics, as detailed in the previous section

$$p_{b_i} = \{\text{insts}|\text{vote}|\text{old}|\text{recent}\} \tag{3}$$

Defects are attached to binary packages, and thus indirectly attached to source packages. Let d be a defect

$$d = \{d_j\}; \quad d_j \in b_i \ \forall b_i \in s \tag{4}$$

This set can be constrained to include only defects subject to some restrictions, such as considering only fixed defects, defects reported in the last month, etc.

The number of defects of a package is the cardinality of the above set

$$n_d = |d| \quad \text{s.t. restrictions} \tag{5}$$

2) Packages Age: The popularity of a package may be affected by the age of the package since older packages have more opportunity to attract new users. To evaluate the influence of the age of packages, we have also calculated their age, using the upload history data.

In contrast to what it is common in free / open source software, when a defect is fixed, or a new version of a package is added to the distribution, changes are not committed to a version control system. The source package is uploaded to the Debian servers, and the binary packages are automatically created for the different architectures in which Debian is available.

We use this upload history to calculate the age of a package. Every upload record is marked with the date of the upload t_u. So the age of a package a can be defined as the difference between the oldest date and the current date t_c

$$a(s) = t_c - \min(t_u) \quad \forall t_u \in s \tag{6}$$

B. Data retrieval

The *Ultimate Debian Database* (UDD) [14] was used to obtain the popularity and defects of every package in Debian (plus some meta-information about the packages). For our analysis, we used the dump of the UDD that was available at its website in the first week of August 2010.

We aggregated the metrics as explained in the previous section, and added that information to the UDD. Our unit of analysis is the *source package*, i.e., we aggregated the data for popularity, defects, etc, for every source package. This allowed for an easy cross correlation of all the metrics and parameters of source packages.

When measuring the defect counts (both fixed and non-fixed), we run into the risk of double counting duplicate reports if users have submitted different reports referring to the same problem. We examined the data for the counts of fixed defects and made sure they do not include duplicate reports and that they refer to unique software issues fixed either by Debian or upstream developers.

With the brief formalization of the analysis shown in the previous subsection, it is easy to extract the necessary data from the UDD. However, we also provide a *replication package* containing all the necessary queries to obtain the data used for this study, and the GNU R [16] code for the plots and statistical tests shown in this paper. The package is available at the permanent URL `http://purl.org/net/who/iht/wcre2011`.

We only considered packages in the stable distribution of Debian, that at the time of writing was the 5.0 release, codename "Lenny".

V. Results

A. Overall analysis

Figure 1 shows a plot of the popularity and number of defect reports for all the source packages in Lenny. Each dot corresponds to a package; for more clarity, the overall trend is shown as a *lowess* line across the dots. The plot is shown in logarithmic scale so very popular and/or defective packages can be compared to the rest of packages. From the figure, it is clear that packages with a high level of popularity have a high number of reported defects. Also, unpopular packages do not have a high number of reported defects.

In the next subsection, we further explore the relationship between popularity and the number of reported defect in more detail, using diverse statistical techniques. Unless otherwise stated, all of our reported results are significant at $p \leq 0.001$.

Correlation Between Defect Reports and Popularity: The Spearman's rank correlation coefficient between defect reports and popularity of a package is 0.45 (significant correlation). We believe that the reason that the correlation is not very strong is due to the high dispersion of the data, making it difficult to extract a clear relationship between the popularity and reported defects.

Comparison of Median Values: Next, we divide the sample into groups, and test whether or not there exists a statistically significant difference between the medians of reported defects. We chose to test the median instead of the mean, because the data is highly skewed.

We divided the sample in five groups, providing an even separation based on the popularity of a package. This way, each quantile (i.e., group) contained the same amount of packages (one fifth of all the packages), which is a required property to be able to apply a statistical tests for difference of the medians. The quantiles are shown in Figure 1, where the vertical dashed line show the values of the different quantiles.

Figure 2 shows a boxplot for each one of the groups. The y-axis represents the number of reported defects in logarithm-scale. Packages in group 5 are the most popular, and packages

in the group 1 are the least popular. The plot shows that the maximum and median value of reported defects is limited by the popularity of the packages. In other words, packages with higher values of popularity have a higher number of reported defects.

Table I shows the values of the mean, median and standard deviation (of the number of reported defects) for each of the groups. We observe that groups with higher popularity have a higher dispersion in the data compared to packages with lower popularity. An in depth investigation of this phenomena showed that the cause of this dispersion is due to the fact that packages with high popularity can have both, a low and high number of reported defects. Packages with low popularity always had a low number of reported defects.

The question that lingers now is whether the difference in reported defects for the different groups is statistically significant or not. To answer this question we performed a Mann-Whitney-Wilcoxon test, comparing the correlative groups. We chose to use the Mann-Whitney-Wilcoxon since it is not sensitive to skewed data, and it does not make any assumption about the shape of the distribution of the data. The results are shown in Table II. The results show that all the differences between the medians of the groups are statistically significant.

TABLE I
DESCRIPTIVE STATISTICS OF THE FIVE GROUPS OF PACKAGES. THE GROUPS ARE PARTITIONED AS IN FIGURE 1.

Group	Mean	Median	St. dev.
1	7	3	14
2	11	6	21
3	17	8	33
4	32	12	60
5	126	31	300

TABLE II
COMPARISON OF THE MEDIANS FOR ALL THE GROUPS USING THE MANN-WHITNEY-WILCOXON TEST.

Groups	W	p-value	Different?
1 – 2	2673386	< 0.001	Yes
2 – 3	4065064	< 0.001	Yes
3 – 4	2889499	< 0.001	Yes
4 – 5	4612536	< 0.001	Yes

Based on our analysis, we conclude that packages with very low popularity will only have a low number of reported defects, and only packages that are very popular will have high number of reported defects.

The number of defects reported for a Debian package is limited by its popularity.

B. Recent Activity

In our previous analysis, we considered the total number of installations (as a measure of popularity) and the number of defects reported during the whole history of Debian. Using all of the reports might be biased. For example, older packages might have had more opportunity to get installed and hence, more reported defects as well. Therefore, in this section we control for age of a package and report our findings.

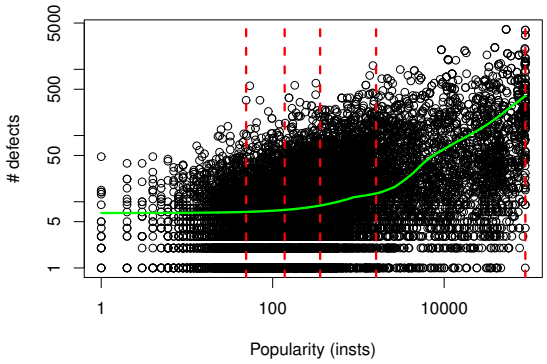

Fig. 1. Popularity of a package against number of defects reported (logarithmic scale). The overall trend line shows that the number of defects grows with the number of installations. The vertical lines show the different groups used in the boxplots.

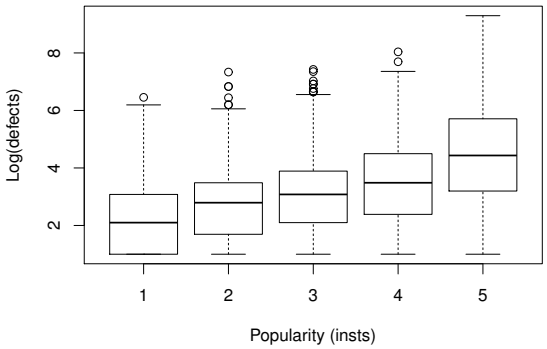

Fig. 2. Popularity of a package against number of defects reported (boxplots). The groups are partitioned as in Figure 1.

In particular we focus on recent popularity data and only defects that have been fixed (and therefore confirmed) recently. The data in the UDD shows that the most recent defect was fixed in August 2010. Also, if we use the "votes" field in the popularity data, we are only considering installations in the last thirty days. Figure 3 shows a plot of the number of defects fixed in 2010, and the number of recent installations (within the last 30 days). The overall trend is shown as a line across the dots. The trend is null for the lower values of popularity, and grows for highly popular packages. Again, the values of the quantiles are shown as vertical dashed lines. These lines divide the sample into five groups that contain the same number of packages[1].

Correlation Between Recent Defect Reports and Recent Popularity: The Spearman's rank correlation is 0.34 (statistically significant), which shows low correlation between the two variables. However, we would like to point out here that there is a drawback to focusing on recent activity. For example, in our data we are counting packages that were not

[1]The number of dots in the plot is not the same though, because a dot can contain several packages with the same pair of values.

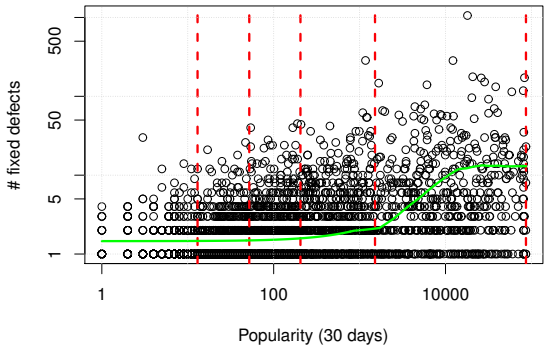

Fig. 3. Popularity of packages (last 30 days) against defects fixed in 2010 (logarithmic scale).

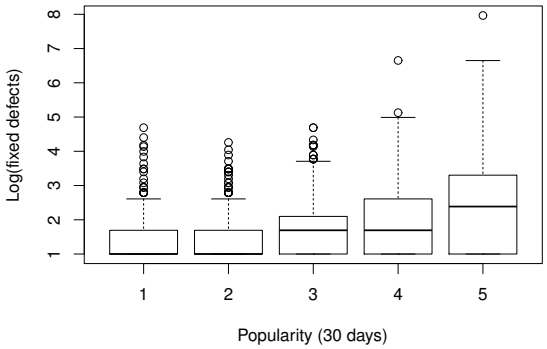

Fig. 4. Popularity of packages (last 30 days) against defects fixed in 2010 (boxplot).

installed at all and that did not get any defect report, which accounts for $1,375$ packages in our sample. Ignoring those packages the coefficient increases to 0.39. There are also some other extreme cases (like packages with no defects and very few installations). Removing those cases would increase the correlation coefficient even more. However, it is difficult to set the threshold to remove data from the sample. Therefore, we decided to repeat the statistical analysis using boxplots and median tests for the recent activity data.

Comparison of Median Values: Figure 4 shows the corresponding boxplots for the five groups. The y-axis shows the number of defect reports fixed during 2010 in logarithm-scale. Although the differences between the quantiles are not as clear as in the previous case, the plot shows that the level of defects for popular packages is much higher than for the least popular packages.

TABLE III
DESCRIPTIVE STATISTICS OF THE FIVE GROUPS OF PACKAGES (RECENT ACTIVITY).

Group	Mean	Median	St. dev.
1	2	1	3
2	2	1	2
3	3	2	4
4	5	2	14
5	13	4	51

Table III shows the values of the descriptive statistics in each group. The first two groups have very similar descriptive statistics. The third group is slightly different, and the fourth and fifth groups are highly dispersed. To see if we can consider these groups to have a different median value from a statistical point of view, we repeat the Mann-Whitney-Wilcoxon test, and present the results in Table IV. The results show that the first two groups are statistically similar, and the rest of groups are different. Only those two last groups have a higher median number of defect reports (fixed during 2010).

In summary, our results show that once again popularity is a limiter for the number of defects that a package will have. This limit is not influenced by age, because in this case all the packages had the same opportunity (in terms of elapsed time) to be installed and to have defects reported against them.

TABLE IV
COMPARISON OF THE SAMPLES USING THE MANN-WHITENY-WILCOXON TEST (RECENT POPULARITY AND FIXED DEFECTS).

Samples	W	p-value	Different?
1 – 2	152267	> 0.001	No
2 – 3	130246.5	< 0.001	Yes
3 – 4	129192.5	< 0.001	Yes
4 – 5	124430	< 0.001	Yes

Even when packages have the same installation time, the number of defect reports for a Debian package is limited by its popularity.

C. Influence of Age on Popularity and Number of Defect Reports

The age of a package can be a cause for bias in the results of our analysis. Figure 5 shows the plot of the age of packages (in number of days) against the popularity of packages (number of installations in the last 30 days). There is a surprising pattern in the data. The number of packages older than $1,000$ days is much higher than younger packages. This phenomena may be explained by the fact that the package integration process in the stable distribution of Debian. When a package is added to Debian, it enters the so-called *unstable distribution*. After some time, it migrates to the *testing distribution*, and if the package is stable enough, after some time it enters the stable distribution. We suspect that this delay to enter the stable distribution is the cause for the observed pattern, but this is something that should be explored.

This issue does not influence our results, since our main hypothesis is that popularity makes packages get more defect reports. If we consider very young packages, which did not have time to develop a mature community of users, the possible lack of popularity and/or defects would not be related to the quality of the package, but to its age.

In any case, besides that surprising pattern (i.e., packages older than 1000 days), there is no relationship between the recent popularity of a package and its age. In fact, the Spearman's rank correlation coefficient for the data shown in Figure 5 is 0.16, which shows a very weak correlation.

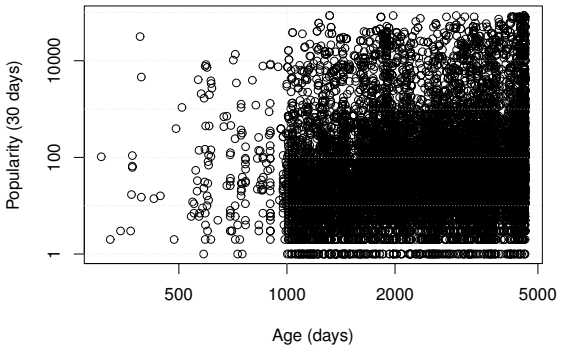

Fig. 5. Age of packages against popularity (last 30 days) in logarithmic scale.

Figure 6 shows the plot of age against the number of defects fixed during 2010. At first glance, contrarily to the previous plot, it seems that there is some relationship between the age and number of fixed defects. However, in this case, the Spearman's rank correlation coefficient is -0.01 and the p value is higher than our threshold. This shows that there is no statistically significant relationship between age and the number of defects fixed during 2010.

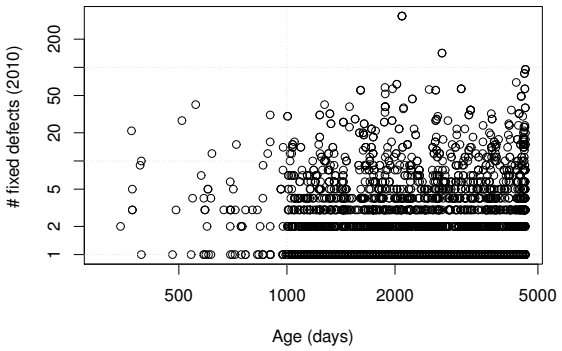

Fig. 6. Age of packages against number of defects fixed during 2010 (logarithmic scale).

The relationship between recent popularity and number of recently fixed defects is not influenced by age.

D. Discrimination by Priority and Type of Package

1) Influence of packages priority: The importance of a package can be a cause for bias in the results of our analysis. All important packages may have high popularity because they are part of the most typical installations of Debian. For instance, all the packages related to the X11 server might appear as highly popular. For source packages, there are different levels of priority in the Debian packaging system: required, standard, important, optional and extra.

To analyze the potential impact of this bias, we divided our sample in two groups: **Important packages** (required, standard or important priorities); and **Non-important** (remaining packages). Figure 7 shows the plot of popularity against defects for the important packages. It is clear that most of

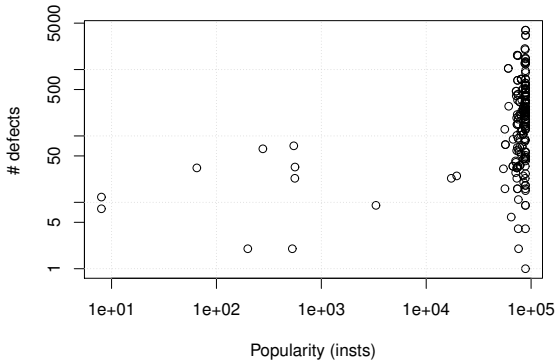

Fig. 7. Popularity of packages against number of defects. Only important packages (logarithmic scale).

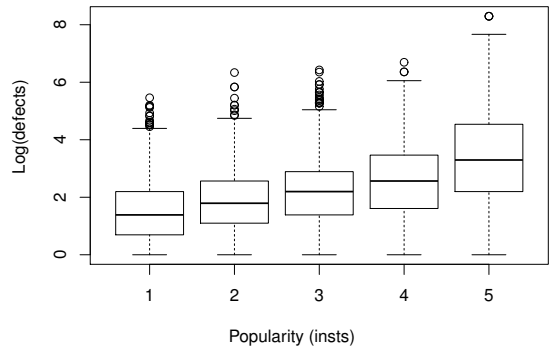

Fig. 8. Popularity of packages against number of defects. Only packages that are voluntarily installed (boxplot).

the important packages are very popular because they are installed by default in the most typical systems. In this case, we cannot really argue about defects and popularity, since users are not voluntarily installing the packages. They do not have the option to uninstall such packages (unless they decide to stop using the whole system).

For the case of packages that are installed optionally, i.e., packages that users voluntarily decide to install, Figure 8 shows the boxplots for overall popularity and number of defect reports. Once again, we observe that more popular packages get a higher number of defect reports. Therefore, the presence of important packages in the sample does not change the observed pattern. The Mann-Whitney-Wilcoxon test shows that the difference between all the groups are statistically significant.

Focusing only on recent and fixed defects, and on recent installations, yields similar results. Figure 9 shows the boxplots of popularity (measured as number of installations in the last 30 days) against number of fixed defects during 2010. Again, we rarely find unpopular packages with a high number of defects.

The Mann-Whitney-Wilcoxon test shows that the first three groups are similar (with the same value of the median), and the fourth and fifth group have a higher median with a difference that is statistically significant.

2) Influence of the Type of Package: One may argue that different types of applications may have different types of users; users' profile may affect the relationship between

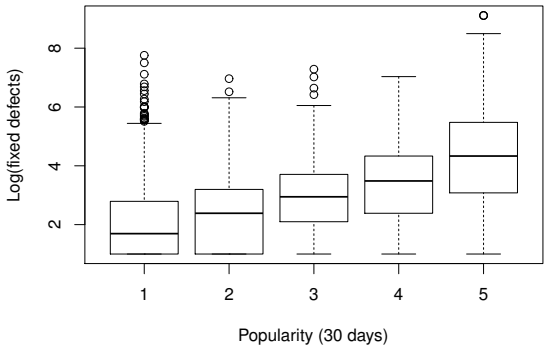

Fig. 9. Popularity of packages (last 30 days installations) against number of fixed defects during 2010. Only packages that are voluntarily installed (boxplot).

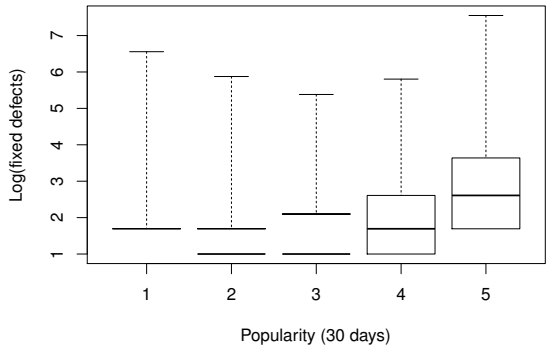

Fig. 11. Popularity (last 30 days) against defects fixed during 2010, for all the defects submitted by Debian developers (boxplots).

TABLE V
SECTIONS OF PACKAGES

Category	Sections	#
Software development	devel, contrib/devel	1046
Libraries	libs, contrib/libs	1273
Utilities	utils, contrib/utils	820
Editors	editors, contrib/editors	117
Games	games, contrib/games	621
Miscellaneous	misc, contrib/misc	378

popularity and number of defect reports. One kind of users can be more prone to submit defect reports than others.

Therefore, we explore how the type of package influences the relationship between popularity and defects. Debian packages contain a field called *section* which can be used to classify packages. There are tens of different sections. We will focus only on some selected categories: software development, libraries, utilities, games, editors and miscellaneous. Table V summarizes the sections included in each category, and the amount of packages per category (non-important packages). We only consider non-important packages, number of recent installations, and defects fixed during 2010.

Figure 10 shows the boxplots of recent installations against defects fixed during 2010, for each one of the considered categories. In each category, we include five boxplots, for each one of the five subsamples obtained by dividing the overall sample using the quantiles, as in the previous cases. In all the cases, the median for the fifth group is statistically different compared to the medians of the rest of the groups (again, using the Mann-Whitney-Wilcoxon test). The rest of the groups have similar or different medians, depending on each case. Therefore, the relationship between popularity and defect reports is verified regardless of the kind of software we are considering.

The type of application and the priority of the package do not influence the relationship between popularity and defects.

E. Defect Report Quality

Defect reports may not all be of the same quality. Some defect reports may be better and more useful for developers than others. Although there are proposals to measure the quality of defect reports [7], [18], here we use a simple approach: we consider defects reported by Debian developers as having a high level of quality.

Defects are reported through email, and all Debian developers have a `debian.org` email account. So if a defect is reported from a Debian email account, we consider it as being reported by a Debian developer. We do not examine other cases, such as Debian developers using a different email account, or upstream developers reporting defects to Debian.

Figure 11 shows the boxplots of popularity (installations in the last 30 days) against the number of defects fixed during 2010, reported by Debian developers. Although the difference in the medians is very small in this case, again using the Mann-Whitney-Wilcoxon test, shows that the median of the fifth and fourth groups (i.e., highly popular packages) are higher than the rest of medians (although the lower popularity groups are more dispersed than in previous cases). The conclusion for defects submitted by users (not using a `debian.org` email address) are similar (not shown here due to space limitations).

Therefore, if we discriminate the defects by reporter, the data still shows that the number of defect reports reported against a package is limited by its popularity. This relationship between popularity and defects is not affected by who reports the defect, i.e., by the quality of the defect report. The behavior is the same for the overall sample and for the sample of defects submitted by Debian developers.

The reporter of a defect report (Debian developer or user) does not influence the relationship between popularity and defect counts.

VI. THREATS TO VALIDITY

Conclusion validity. All the results shown here were extracted with statistical analysis that were significant at the $p \leq 0.001$ level. To compare the medians, we have used the Mann-Whitney-Wilcoxon test, which is robust in what regards to

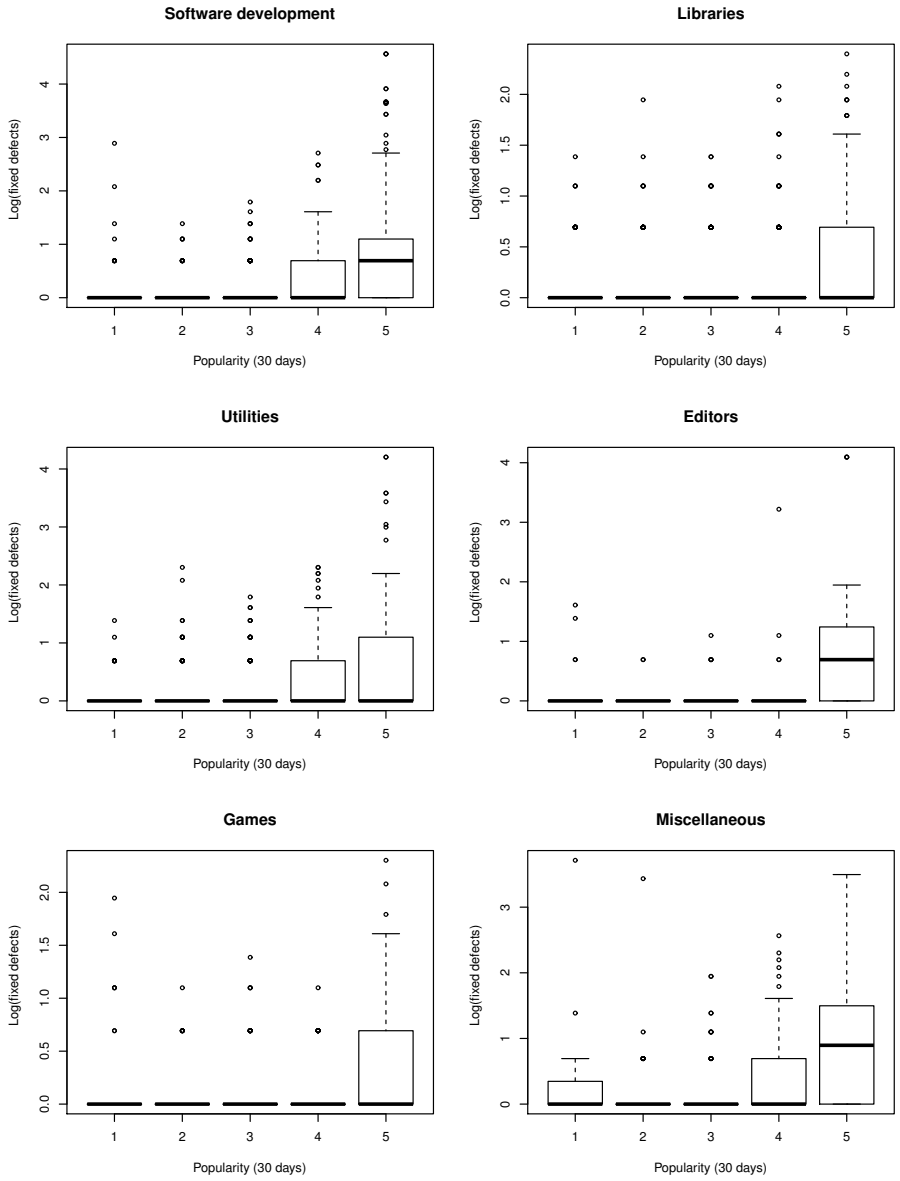

Fig. 10. Popularity (30 days) against fixed defects during 2010, for the different sections of packages. Only packages that are voluntarily installed (boxplots).

the statistical distribution of the data. Also, all the tests were performed using samples of the same size.

On the other hand, the conclusion validity of this study depends on the validity of the *Ultimate Debian Database* [14], as we did not collect any data ourselves for this study, or applied any measurement or instrumentation directly.

External validity. Our analysis is based only on data extracted from Debian, through the Ultimate Debian Database. It is unclear whether the relationship between popularity and defects can be extracted for other open source software. This influences the kind of defects and users that are considering for our analysis. Defects that are fixed outside Debian can be of a different nature. We are only considering a subclass of all kinds of defects that can be found in an free / open source software development. To overcome this threat, the study should be extended to other cases.

Internal validity. We are also considering that every application is independent with regards to the measured variables (defects, popularity). It could happen that the most frequently installed applications are also the most complex, and that this complexity is the cause for a higher number of defect reports. To address this threat, we should extend our study with additional packages metrics, to control for other variables, such as software complexity or packages dependencies.

Construct validity. The popularity of packages is another source of threats to the validity of this analysis. Only users that have voluntarily opted-in can submit data to the Popcon server. If there is any relationship between the users' profile and the reasons to opt-in (or opt-out), then we are only counting a particular kind of users. However, if such relationship does not exist, we can assume that users that submit information

to the Popcon servers are a representative subsample of the overall population of Debian users.

This threat to the validity of our analysis can be solved by gathering more information about the popularity of open source systems using surveys. This approach would provide even richer information about popularity, which is very limited in the case of Debian (only number of installations, discriminating between old and recent events).

VII. Conclusions

In software defects research, it is often assumed that a higher number of defect reports corresponds to a higher number of actual defects. In this paper, we conduct a case study on the Debian GNU/Linux distribution to study the relationship between defects and popularity, measuring the popularity and number of defects for a sample of more than $13,000$ packages included in the Lenny release of Debian.

Our analysis shows that it is very difficult to find highly defective packages if their user base is low. In other words, only very popular packages contain a high level of reported defects. The lack of defect reports can be related to a low adoption of a program.

We believe that our finding have significant implications on software defect studies based on free / open source software data. We cannot assume that a high level of reported defects is related to the quality of a system. More popular software will have more defect reports, as there are more users to discover defects. Future software quality studies should consider the type of bias we found in this study, which we call *popularity bias*.

As further work, we plan to extend this analysis with more variables, such as package size and complexity, to determine if the defects variable can be better explained by other variables instead of complexity. Furthermore, we will assess how defects predictor models are affected by this *popularity bias*. The performance of the models can be affected when taking into account the influence of popularity or installation counts; we think that models which only predict the presence or absence of defects (boolean models) will be less sensitive than models predicting defects counts. Finally, we also plan to explore how to obtain data about popularity for other cases in practice.

VIII. Acknowledgments

Part of this work was done while Israel Herraiz was visiting the *Software Analysis and Intelligence Lab* of *Queen's University*, thanks to the funding provided by the *Fundación Universidad Alfonso X el Sabio* and *Banco Santander*.

References

[1] G. Antoniol, K. Ayari, M. D. Penta, F. Khomh, and Y.-G. Guhneuc, "Is it a bug or an enhancement?: a text-based approach to classify change requests," in *Proceedings of the Conference of the Center for Advanced Studies on Collaborative Research: Meeting of Minds*. Ontario, Canada: ACM, 2008, pp. 304–318.

[2] C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, and P. Devanbu, "Fair and balanced?: bias in bug-fix datasets," in *Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*. ACM, 2009, pp. 121–130.

[3] M. Cataldo, A. Mockus, J. A. Roberts, and J. D. Herbsleb, "Software dependencies, work dependencies, and their impact on failures," *IEEE Transactions on Software Engineering*, vol. 35, no. 6, pp. 864–878, 2009.

[4] J. Davies, H. Zhang, L. Nussbaum, and D. German, "Perspectives on bugs in the debian bug tracking system," in *7th IEEE Working Conference on Mining Software Repositories (MSR)*, May 2010, pp. 86 –89.

[5] N. E. Fenton and N. Ohlsson, "Quantitative analysis of faults and failures in a complex software system," *IEEE Transactions on Software Engineering*, vol. 26, no. 8, pp. 797–814, August 2000.

[6] T. Gyimothy, R. Ferenc, and I. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction," *IEEE Transactions on Software Engineering*, vol. 31, no. 10, pp. 897 – 910, 2005.

[7] P. Hooimeijer and W. Weimer, "Modeling bug report quality," in *Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering*. ACM, 2007, pp. 34–43.

[8] A. Meneely, L. Williams, W. Snipes, and J. Osborne, "Predicting failures with developer networks and social network analysis," in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*. ACM, 2008, pp. 13–23.

[9] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Transactions on Software Engineering*, vol. 33, no. 1, pp. 2–13, January 2007.

[10] M. Michlmayr and A. Senyard, "A statistical analysis of defects in Debian and strategies for improving quality in free software projects," in *The Economics of Open Source Software Development*, J. Bitzer and P. J. H. Schrder, Eds. Amsterdam, The Netherlands: Elsevier, 2006, pp. 131–148.

[11] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Two case studies of Open Source software development: Apache and Mozilla," *ACM Transactions on Software Engineering and Methodology*, vol. 11, no. 3, pp. 309–346, 2002.

[12] A. Mockus, P. Zhang, and P. L. Li, "Predictors of customer perceived software quality," in *Proceedings of the 27th International Conference on Software Engineering (ICSE)*. ACM, 2005, pp. 225–233.

[13] T. H. D. Nguyen, B. Adams, and A. E. Hassan, "A case study of bias in bug-fix datasets," in *Working Conference on Reverse Engineering*, Beverly, Massachusetts, 2010.

[14] L. Nussbaum and S. Zacchiroli, "The Ultimate Debian Database: Consolidating Bazaar Metadata for Quality Assurance and Data Mining," in *7th IEEE Working Conference on Mining Software Repositories (MSR)*, 2010.

[15] J. W. Paulson, G. Succi, and A. Eberlein, "An empirical study of open-source and closed-source software products," *IEEE Transactions on Software Engineering*, vol. 30, no. 4, April 2004.

[16] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2011, ISBN 3-900051-07-0. [Online]. Available: http://www.R-project.org/

[17] G. Robles, J. M. Gonzalez-Barahona, M. Michlmayr, J. J. Amor, and D. M. German, "Macro-level software evolution: A case study of a large software compilation," *Empirical Software Engineering*, vol. 14, no. 3, pp. 262–285, June 2009.

[18] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss, "What makes a good bug report?" *IEEE Transactions on Software Engineering*, vol. 36, no. 5, pp. 618 –643, 2010.

[19] T. Zimmermann and N. Nagappan, "Predicting defects using network analysis on dependency graphs," in *Proceedings of the 30th International Conference on Software Engineering (ICSE)*. Leipzig, Germany: ACM, 2008, pp. 531–540.

[20] T. Zimmermann, R. Premraj, and A. Zeller, "Predicting defects for Eclipse," in *Proceedings of the 3rd International Workshop on Predictor Models in Software Engineering (PROMISE)*. IEEE Computer Society, 2007.